Simons

Eine notwendige Erweiterung für den Commodore 64

Erfahrene Commodore-Programmierer werden mir sicherlich zustimmen: Die ausgezeichneten Commodore-Editiermöglichkeiten verhalten sich für den Programmierer umgekehrt proportional zu den Basic-Versionen. Dies ist auch bei dem Commodore 64 nicht anders. Obwohl der C 64 über eine hochauflösende Grafik verfügt, bietet das Standard-Basic hier keine Unterstützung. Für häufige Programmierarbeiten sind Basic-Erweiterungen — insbesondere für den Grafikteil und die Sprites — eine notwendige Hilfe. Eine solche Erweiterung ist Simons Basic für den Commodore 64.

Teil 1

s Basic

Simons Basic bietet sehr viele wichtige Befehle. Bild 1 zeigt eine Übersicht über alle Befehle und eine Kurzbeschreibung ihrer Bedeutung. Diese Übersicht kann auch als Handzettel für diejenigen dienen, die schon mit Simons Basic arbeiten.

Simons Basic enthält viele dringend notwendige Befehle, aber auch Befehle, die wohl nur in sehr seltenen Fällen benutzt werden. Auf jeden Fall ist Simons Basic für den geübten Programmierer eine wertvolle Unterstützung. Besonders hervorzuheben sind hier die Befehle, die in Bild 1 unter Programmierhilfen zusammengestellt sind, die in dieser oder ähnlicher Form auch schon von anderen Programmier-Kits her bekannt sein dürften. Weiterhin einige Befehle zur Verarbeitung von Zeichenreihen wie zum Beispiel INST. Für Programmierer, die auch andere Programmiersprachen wie PL/1 oder Pascal kennen, dürften besonders die neuen Strukturbefehle und die ERROR-Befehle interessant sein.

Um die speziellen Möglichkeiten des Commodore 64 wie die hochauflösende Grafik, die Definition von Sprites und den Sound-Generator zu benutzen, sind natürlich die entsprechenden Befehle notwendige Voraussetzung, wenn Programmieren nicht in Byte-Fummelei ausarten soll.

Zu den Befehlen, die wohl nur dann angewendet werden, wenn ein Programmierer auch alle Register des Computer sehen will, gehören neben einigen Befehlen aus den anderen Bereichen bestimmt auch alle Befehle der Bildschirmsteuerung.

Alles in allem kann man jedoch sagen: Der zusätzliche Befehlsvorrat von Simons Basic läßt fast keine Wünsche offen.

Gehen wir im folgenden kurz auf die verschiedenen Befehle und ihre Anwendungsmöglichkeiten ein:

Programmierhilfen

AUTO

Dieser Befehl ist von anderen Kits bestimmt schon hinlänglich bekannt. Er ermöglicht die zeilenweise Programmeditierung, ohne jeweils eine neue Zeilennummer miteintippen zu müssen. Dies erspart hauptsächlich beim fließenden Eintippen eines Programms die Überlegung: Welches ist denn jetzt die nächste Zeile?

COLD

Dieser Befehl ersetzt das Ein- und Ausschalten des Computers, wenn ein Kaltstart durchgeführt werden soll. Intern werden im Computer immer Zeiger verwaltet, die auf den Anfang des Programms, den Anfang der Variablenbereiche und so weiter zeigen. Der Befehl COLD bewirkt nichts anderes als das Rücksetzen dieser Zeiger in den Ausgangszustand.

DELAY

Mit dem Befehl DELAY kann die Listgeschwindigkeit eingestellt werden. Daß hier 256 Möglichkeiten zur Verfügung stehen, ist mehr als ein Programmierer benötigt. Prinzipiell wird sich jeder aus den Möglichkeiten ein oder zwei Geschwindigkeiten aussuchen, die seiner Lesegeschwindigkeit am Bildschirm entsprechen.

DISAPA

In Verbindung mit dem Befehl SECURE ist der Befehl DISAPA ein hinreichend wirkungsvolles Mittel, um sein Programm gegen unbefugtes Auflisten zu schützen. Im Prinzip wäre es möglich, das gesamte Programm mit diesem Befehl zu schützen, jedoch macht man sich selbst die Arbeit der Softwarepflege damit nur schwieriger. Sinnvoll wäre es, diesen Befehl in einem kurzen Programmstück zu verwenden, welches einige andere Sicherungsmethoden enthält.

DISPLAY

Eine reine Informationsanweisung, die aber sehr wichtig ist, da es sonst sehr schwierig wäre, die aktuelle Belegung der Funktionstasten festzustellen.

DUMP

Der Vorteil eines Interpreters liegt zu einem großen Teil darin, daß während eines Programmlaufes das Programm abgebrochen werden kann und die Variablen abgefragt werden können. Dies erleichtert das Austesten erheblich gegenüber Compilerversionen. Nun ist es recht mühsam, immer nach einem BREAK im Programm einen PRINT-Befehl für alle — oder auch nur die benötigten — Variablen einzugeben, wenn mehrere sogenannte BREAK-POINTS gesetzt sind. Diese Arbeit erleichtert der DUMP-Befehl.

FIND

Ähnlich dem DUMP-Befehl erleichtert der FIND-Befehl das Testen sowie das Dokumentieren von Programmen. Besonders bei langen Listings ist es sehr mühsam, das gesamte Programm nach einer bestimmten Variablen zu durchsuchen. Da in Basic auch im Prinzip alle Variablen global sind, dürften — außer temporären Variablen — den Variablen nicht mehrfache Bedeutungen zugewiesen werden. Mit dem FIND-Befehl ist es unter anderem möglich zu prüfen, ob eine Variable schon im Programm vorhanden ist oder nicht.

KEY

Da der Commodore 64 Funktionstasten anbietet, ist es auch sinnvoll, diese mit häufig verwendeten Basic-Befehlen (zum Beispiel LIST) zu belegen.

Simons Basic

MERGE

Der MERGE-Befehl ermöglicht zwar das Einkopieren von anderen Programmen in ein Programm, das sich im Hauptspeicher befindet, jedoch läßt dieser Befehl einige Möglichkeiten vermissen. Zum Beispiel ist das Laden von bestimmten Programmteilen eines Programms von Diskette nicht möglich. Dies ist besonders dann ein Nachteil, wenn aus anderen Programmen nur bestimmte Unterprogramme übernommen werden sollen.

OLD

Ab und zu kann es vorkommen, daß versehentlich ein NEW-Befehl eingegeben wurde, und man feststellt, daß das Programm vorher nicht abgespeichert war beziehungsweise die Kontrollampe an dem Floppy Disk-Laufwerk blinkt. Da durch den NEW-Befehl nur Zeiger intern umbesetzt werden, ist eigentlich noch nicht alles verloren. Aber es ist doch sehr mühsam, das Programmende des Programms und die Werte für den Beginn der Variablentabelle und so weiter ausfindig zu machen. Dies erspart einem der OLD-Befehl.

OPTION

Eine Anwendungsmöglichkeit für diesen Befehl, der alle Befehle von Simons-Basic hervorhebt, ist direkt nicht ersichtlich. Nützlich ist er vielleicht, wenn ein Programm in normales Basic umgeschrieben werden soll. Aber wenn jemand ein Programm, das mit Simons Basic erstellt wurde, erhält, und dies umschreiben will, weil ihm die Programmierunterstützung nicht zur Verfügung steht, der könnte diesen Befehl gebrauchen. Aber der hat ja kein Simons Basic. Und wer gibt schon seine Programme weiter mit einer Liste: Hier sind die Befehle, die geändert werden müssen?

PAGE

Da der Bildschirm des Commodore 64 nur 40 Zeichen je Zeile hat und das Auslisten der Programme sehr schnell geht, verschwinden Programmstücke nach oben aus dem Bildschirm heraus schneller, als man eventuell die STOP-Taste gefunden hat. Dies kann man einerseits mit der Benutzung der CTRL-Taste beeinflussen, andererseits mit dem weiter vorn beschriebenen DELAY-Befehl. Komfortabel ist es natürlich, wenn man vor Beginn

Befehlsübersicht Simons Basic:

Programmierhilfen:

AUTO	—	Zeilennummernvergabe bei Programmeditierung
COLD	—	Kaltstart, ersetzt aus-/einschalten
DELAY	—	Listgeschwindigkeit einstellen
DISAPA	—	Anweisung schützen
DISPLAY	—	Belegung der Funktionstasten anzeigen
DUMP	—	Variablen mit Werten anzeigen
FIND	—	Basic-Befehle oder Zeichenreihen im Programm suchen
KEY	—	Funktionstasten mit Basicbefehl belegen
MERGE	—	anderes Programm in bestehendes einkopieren
OLD	—	NEW-Befehl aufheben
OPTION	—	Simons Basicbefehle hervorheben
PAGE	—	seitenweise Listenausgabe
RENUMBER	—	Zeilen umnumerieren (ohne Zeilenangaben bei GOTO und GOSUB)
SECURE	—	Programmzeile schützen
TRACE/RETRACE	—	aktuelle Zeilennummer, die im Programm durchlaufen wird, anzeigen und wieder aufheben

Struktur-Befehle und ERROR-Befehle:

CALL	—	Sprung zu einer mit PROC definierten Routine (ähnlich GOTO)
END PROC	—	Ende einer Routine, ähnlich RETURN
EXEC	—	Unterprogrammaufruf für Routinen die mit PROC und END PROC definiert wurden
GLOBAL	—	ursprünglichen Variablenwert wieder zuweisen
IF..THEN..ELSE	—	Bedingte Anweisung mit doppelter Anwendungsmöglichkeit
LOCAL	—	Block bedingte Variablen
LOOP..EXIT IF ..END LOOP	—	Schleifendurchlauf mit bedingtem Abbruch
NO ERROR	—	Fehlermeldung unterdrücken
ON ERROR	—	Sprungverteile für Fehlermeldungen
PROC	—	Sprungadresse (symbolisch)
RCOMP..ELSE	—	Bedingte Anweisung, wobei die Bedingung von der letzten IF-Abfrage übernommen wird
REPEAT..UNTIL	—	ähnlich FOR..NEXT für bedingte Schleifen

Grafik-Befehle

ANGL	—	Radius zeichnen
ARC	—	Segment zeichnen
BLOCK	—	farbig ausgefülltes Rechteck ausgeben
CHAR	—	Zeichen in Grafik-Bildschirm
CIRCLE	—	Ellipse (Sonderfall: Kreis) ausgeben
CSET	—	Zeichensatz umschalten
DRAW	—	Figur zeichnen
HICOL	—	Nach LOW COL zum zurücksetzen auf die drei Farben, die mit MULTI definiert werden
HIRES	—	hochauflösende Grafik (mit Wahl der Vordergrund- und Hintergrundfarbe) einschalten
LINE	—	Linie zeichnen
LOW COL	—	drei weitere Farben zum Multi-Color-Modus zuschalten
MULTI	—	Multi-Color-Modus mit drei Zeichenfarben bestimmen
PAINT	—	Fläche mit Farbe füllen
PLOT	—	Punkt ausgeben
REC	—	Rechteck zeichnen
ROT	—	Figur drehen
TEST	—	Punkt vorhanden?
TEXT	—	Text in Grafik-Bildschirm

Sprite-Befehle:

(KLAMMERAFFE)	—	Form eines Sprites definieren
CHECK	—	Kollision abfragen
CMOB	—	Farben für Multi-Color-Sprite festlegen
DESIGN	—	Speicherzuteilung für Sprite
DETECT	—	Kollision vorbereiten
MMOB	—	Sprite darstellen oder bewegen
MOB OFF	—	Sprite ausschalten
MOB SET	—	Eigenschaften eines Sprite festlegen
RLOCMOB	—	Sprite bewegen

einer jeden Programmiersitzung den Befehl Page verwendet, womit ein seitenweises Blättern in Vorwärtsrichtung erzielt werden kann.

RENUMBER

Wo fast jedes auf dem Commodore 64 erstelltes Programm dynamisch wächst, wird mal hier eine Zeile eingefügt, mal wird dort eine Zeile herausgenommen. Um dieses ganze Zeilennummern-Wirrwarr in den Griff zu bekommen ist natürlich der RENUMBER-Befehl sehr nützlich. Leider wirkt sich der RENUMBER-Befehl nicht auf solche Zeilennummern aus, die hinter GOTO und GOSUB stehen. In mühsamer Kleinarbeit artet es dann aus, wenn Sie anschließend alle Sprungadressen bei GOTO/GOSUB-Befehl von Hand ändern müssen.

SECURE

Dieser Befehl bewirkt nur das eigentliche Schützen, der durch den Befehl DISAPA gekennzeichneten Befehle.

TRACE/RETRACE

Zum Testen von Programmen — besonders bei sogenannten Endlos-Schleifen — leistet der TRACE-Befehl, mit dem die aktuelle Zeilennummer eines laufenden Programmes angezeigt wird, sehr nützliche Hilfe.

Strukturbefehle und ERROR-Befehle

Diese Befehle lassen sich als Einzelbefehle nicht ausreichend erklären, da sie eine gewisse Blockstruktur voraussetzen, so daß wir diese im Zusammenhang besprechen wollen.

Schleifen/bedingte Schleifen/ bedingte Anweisungen

Der erste Bereich der Strukturbefehle widmet sich den Schleifen und bedingten Anweisungen. Da das normale Basic nur IF...THEN-Befehle zuläßt, ist es eine wesentliche Vereinfachung, wenn diese Befehle auch einen ELSE-Teil erhalten. Dadurch können aufwendige Konstruktionen mit GOTO-Befehl vermieden werden, wie Bild 2 zeigt. Bild 3 zeigt die Bedingungen bei einem IF-Statement, die sehr komplex sein können, so daß es sinnvoll ist, diese Bedingung in einem weiteren Befehl ohne erneute Eingabe wieder prüfen zu können. Dies kann mit dem Befehl

Bild 1. Diese Befehle bietet Simons Basic

Musik-Befehle:

ENVELOPE	—	Hüllkurve einstellen
MUSIC	—	Noten festlegen
PLAY	—	Musikwiedergabe
VOL	—	Lautstärke einstellen
WAVE	—	Wellenform einstellen

Befehle für Zeichenreihen

AT	—	Zeichenreihe auf Bildschirm positionieren
CENTRE	—	Ausgabe einer Zeichenreihe in der Mitte einer Bildschirmzeile
CHAR	—	Zeichen in Grafik-Bildschirm
DUP	—	Zeichenreihe vervielfachen
INSERT	—	Zeichenreihe in andere einfügen
INST	—	Zeichenreihe mit einer anderen überschreiben
PLACE	—	Zeichenreihe in Zeichenreihe suchen
TEXT	—	Text in Grafik-Bildschirm

Befehl für Zahlen:

$	—	Umwandlung Hexadezimal in Dezimal
%	—	Umwandlung Binär in Dezimal
DIV	—	Division ohne Rest
EXOR	—	bitweise Verknüpfung von Zahlen mit EXKLUSIV ODER
FRAC	—	Nachkommastellen einer Dezimalzahl

Bildschirmsteuerung

BFLASH	—	Farbwechsel Bildschirmrahmen einschalten
BFLASH O	—	Farbwechsel Bildschirmrahmen ausschalten
COPY	—	Hardcopy einer hochauflösenden Grafik
DOWN	—	Bildschirmbereich nach unten rollen
FCHR	—	Bildschirmbereich mit Zeichen füllen
FCOL	—	Zeichenfarbe in Bildschirmbereich bestimmen
FLASH	—	Blinken einer Bildschirmfarbe einschalten
FILL	—	Bildschirmbereich mit Farbe und Zeichen füllen
HRDCPY	—	Hardcopy eines normalen Bildschirmes
INV	—	Bildschirmbereich invertieren
LEFT	—	Bildschirmbereich nach links rollen
MOVE	—	Bildschirmbereich duplizieren
OFF	—	Blinken einer Bildschirmfarbe ausschalten
RIGHT	—	Bildschirmbereich nach rechts rollen
SCRLD	—	Bildschirm (der mit SCRSV gespeichert wurde) laden
SCRSV	—	Bildschirm (Normal-Modus) speichern
UP	—	Bildschirmbereich nach oben rollen

Befehle für Light-Pen, Joystick und Paddle

JOY	—	Funktion des Joystick bestimmen
PENX	—	X-Koordinate des Light-Pen
PENY	—	Y-Koordinate des Light-Pen
POT	—	Widerstand Paddle feststellen (Potentiometer)

Sonstige Befehle

(KLAMMERAFFE)	—	neues Zeichen definieren
DESIGN	—	neu zu erstellendes Zeichen festlegen
DIR	—	Inhaltsverzeichnis einer Diskette ganz oder teilweise (Jokerzeichen) anzeigen
DISC	—	Diskbefehl ausführen
FETCH	—	Kontrollierte Eingabe
INKEY	—	Abfrage auf gedrückte Funktionstaste
LIN	—	aktuelle Zeile des Cursors anzeigen
MEM	—	Zeichensatz von ROM-Bereich in RAM-Bereich verlegen
PAUSE	—	Pause im Programm (ersetzt »leere« FOR...NEXT—Schleife)
RESET	—	Zeiger auf beliebige DATA-Zeile setzen

Simons Basic

RCOMP...ELSE, da nicht wie in anderen Programmiersprachen eine blockweise Bearbeitung in verschiedenen Zeilen der THEN-/ELSE-Teile erfolgen kann, substituiert werden.

Eine weitere Verbesserung ist die Programmierung von Schleifen mit Bedingungsteilen. Das Beispiel im Handbuch ist relativ ungünstig gewählt, da dieses Beispiel durch eine einfache FOR...NEXT-Schleife ersetzt werden kann. Bild 4 zeigt einen sinnvollen Einsatz für den Befehl REPEAT...UNTIL. Dabei wird die Schleife abgebrochen, wenn eine Bedingung erfüllt ist, die nicht in einer FOR...NEXT-Schleife einprogrammiert werden kann. Sicherlich ist es auch bei einfachen FOR....NEXT-Schleifen möglich, diese Schleifen mit einer IF-Abfrage zu verlassen, jedoch wird das Programm durch die neuen Befehle viel übersichtlicher. Ähnliches leistet auch der Befehl LOOP...EXIT IF...END LOOP.

Prozeduren

Sehr schön handhaben läßt sich die Verwendung von Unterprogrammen als Prozeduren mit Simons Basic. Wie in blockorientierten Sprachen existiert auch ein Befehl PROC, der praktisch die Marke eines Unterprogrammes ist. Das Unterprogramm (in diesem Fall spricht man von Prozedur) wird auch nicht mit RETURN beendet, sondern mit END PROC. Der Aufruf kann mit CALL oder mit EXEC erfolgen, wobei CALL einem GOTO entspricht (eine unübliche Art des Aufrufs einer Prozedur, da Prozeduren normal unabhängig von ihrer Lage im Programm ausgeführt werden) und EXEC einem GOSUB.

```
100 REM OHNE IF...THEN...ELSE
110 IF A=B THEN C=D : GOTO 130
120 E=F
130 REM FORTSETZUNG
140 :
150 :
160 :
170 REM MIT IF...THEN...ELSE
180 IF A=B THEN C=D ELSE E=F
190 REM FORTSETZUNG
```

Bild 2. Ein Beispiel für IF...THEN...ELSE

```
 90 REM OHNE RCOMP...ELSE
 100 IF A=B AND X=Y OR F<G AND H>J AND NOT Y=R THEN PRINT "SEHR LANGER TEXT";
 110 IF A=B AND X=Y OR F<G AND H>J AND NOT Y=R THEN PRINT "DER NICHT IN EINE";
 120 IF A=B AND X=Y OR F<G AND H>J AND NOT Y=R THEN PRINT "ZEILE PASST. !!!!";
 130 IF A=B AND X=Y OR F<G AND H>J AND NOT Y=R THEN GOTO 150
 140 PRINT"NOCH EIN TEXT"
 150 REM FORTSETZUNG
 160 :
 170 :
 180 :
 190 REM MIT RCOMP...ELSE
 200 IF A=B AND X=Y OR F<G AND H>J AND NOT Y=R THEN PRINT "SEHR LANGER TEXT";
 210 RCOMP PRINT"DER NICHT IN EINE ZEILE PASST !!!!!" ELSE PRINT"NOCH EIN TEXT
 220 REM FORTSETZUNG
READY.
```

Bild 3. Beispiel für RCOMP...ELSE

```
100 REM VERGLEICH ZWEIER ZAHLEN ALS ABBRUCHKRITERIUM
110 REPEAT
120 ZN = ZA / 3
130 UNTIL ABS(ZN-ZA) <0.0000001
```

Bild 4. So wird REPEAT...UNTIL eingesetzt.

Wenn auch keine Blockvariablen im ursprünglichen Sinne zugelassen sind, kann man jedoch mit dem Befehl LOCAL Variableninhalte retten und später mit dem Befehl GLOBAL wieder auf diese Werte zurückgreifen. Dies erleichtert insbesondere die Programmierung großer komplexer Programme mit vielen Prozeduren.

Fehlerbehandlung

Die beiden Befehle ON ERROR und NO ERROR erlauben eine relativ komfortable Fehlerbehandlung. Die normale Fehlerbehandlung (Programmabbruch mit Anzeige des Fehlers) ist in den meisten Fällen nicht sehr benutzerfreundlich, da die Fehler per Programm abgefangen und durch eine entsprechende Benutzermitteilung eventuell auch behoben werden könnten. Mit dem Befehl ON ERROR ist eine solche komfortable Fehlerbehandlung in Abhängigkeit des aufgetretenen Fehlers (Liste im Handbuch enthalten) möglich. Lediglich eine Unterdrückung der Fehlermeldungen ist durch den Befehl NO ERROR möglich.

In der nächsten Ausgabe werden wir uns mit den Grafik-, Sprite- und Musik-Befehlen von Simons Basic sowie mit den Befehlen für Zeichenreihen, Zahlen, Light-Pen, Joystick, Paddle und der Bildschirmsteuerung beschäftigen.

(H.L. Schneider)